In the Claims

Please amend Claims 1, 8, and 15, cancel Claims 16-17, and add Claims 25-26, as shown below. Applicant respectfully reserves the right to prosecute any originally presented claims in a continuing or future application.

1. (Currently Amended) A system that provides a generic user interface testing framework, and allows a user to test and debug graphical user interfaces for software applications under development, comprising:

a computer including a computer readable medium, and a processor operating thereon;

a software application source code, stored on the computer readable medium, wherein the software application source code defines a software application under development, including a graphical user interface as part of the software application, and wherein the software application source code executes on the computer to display its graphical user interface;

a plurality of different software test tools, wherein each software test tool is operable to test a plurality of different graphical user interfaces for a plurality of different software applications and wherein each software test tool is associated with a different tool-specific scripting language, that can be invoked by a user to perform testing operations on the graphical user interface that is displayed while the software application is running, and wherein each of the plurality of different software test tools use only their associated tool-specific scripting language to test the plurality of different graphical user interfaces associated with [[a]] the plurality of different software applications;

a test case input file stored on the computer readable medium, that contains a plurality of generic interface commands that are abstractions independent of any of the tool-specific scripting languages, wherein the test case input file can be edited and reused as necessary by the user to specify different generic interface commands for testing against a software application's graphical user interface in any of the different software test tools; and

an interpretive engine that executes on the computer, and that includes a plurality of dynamically loaded libraries corresponding to the plurality of different software test tools, and including [[a]] at least one library for each of the plurality of different software test tools wherein each library is a group of functions written in each tool-specific scripting language, wherein the interpretive engine receives the generic interface commands defined in the test case input file, identifies which libraries are required, loads the required libraries associated with the software test tool the user is currently using, maps the generic interface commands to the software test tool's associated tool-specific scripting language, uses the software test tool to perform the

testing operations on the software application's graphical user interface using the associated tool-specific scripting language, and reports to the user the success or failure of the testing operations.

2. (Previously Presented) The system of claim 1 wherein the system includes the software test tools stored locally on a computer processing system containing the user interface testing framework.

3. (Previously Presented) The system of claim 1 wherein software test tools are stored at another computer processing system or machine.

4. (Previously Presented) The system of claim 1 further comprising a rules-based wizard that guides the user to edit or create the test case input file by choosing the testing operations to be included in the test case input file wherein the rules-based wizard maps the testing operations to generic interface commands.

5. (Canceled).

6. (Previously Presented) The system of claim 1 wherein the test case input file is created offline and subsequently communicated to the interpretive engine.

7. (Previously Presented) The system of claim 1 wherein any of the software test tools can be removed and replaced with another software test tool.

8. (Currently Amended) A method for providing a generic user interface testing framework that allows a user to test and debug graphical user interfaces for software applications under development, comprising the steps of:

executing a software application source code stored on a computer readable medium, wherein the software application source code defines a software application under development, including a graphical user interface as part of the software application, and wherein the software the software application source code executes to display its graphical user interface;

providing a plurality of different software test tools, wherein each software test tool is operable to test a plurality of different graphical user interfaces for a plurality of different

software applications and wherein each software test tool is associated with a different tool-specific scripting language, that can be invoked to perform testing operations on the graphical user interface that is displayed while the software application is running, and wherein each of the plurality of different software test tools use only their associated tool-specific scripting language to test the plurality of different graphical user interfaces associated with a plurality of different software applications;

allowing a user to enter a test case input file stored on the computer readable medium, that contains a plurality of generic interface commands that are abstractions independent of any of the tool-specific scripting languages, wherein the test case input file can be edited and reused as necessary by the user to specify different generic interface commands for testing against a software application's graphical user interface in any of the different software test tools; and

using a plurality of dynamically loaded libraries corresponding to the plurality of different software test tools, and including [[a]] at least one library for each of the a plurality of different software test tools wherein each library is a group of functions written in each tool-specific scripting language, to receive the generic interface commands defined in the test case input file, identify which libraries are required, load the required libraries associated with the software test tool the user is currently using, map the generic interface commands to the software test tool's associated tool-specific scripting language, use the software test tool to perform the testing operations on the software application's graphical user interface, including translating the generic interface commands to tool-specific commands, and report to the user the success or failure of the testing operations.

9. (Previously Presented) The method of claim 8 wherein the software test tools are stored locally on a same computer or machine as the software application under development.

10. (Previously Presented) The method of claim 8 wherein the software test tools are stored at another computer or machine as the software application under development.

11. (Previously Presented) The method of claim 8 further comprising a rules-based wizard that guides the user to edit or create the test case input file by choosing the testing operations to be included in the test case input file wherein the rules-based wizard maps the testing operations to generic interface commands.

12. (Canceled).

13. (Previously Presented) The method of claim 8 wherein the test case input file is created offline and subsequently communicated to the interpretive engine.

14. (Previously Presented) The method of claim 8 wherein any of the software test tools can be removed and replaced with another software test tool.

15. (Currently Amended) A computer readable medium including instructions stored thereon which when executed cause the computer to perform the steps of:

executing a software application source code stored on a computer readable medium, wherein the software application source code defines a software application under development, including a graphical user interface as part of the software application, and wherein the software the software application source code executes to display its graphical user interface;

providing a plurality of different software test tools, wherein each software test tool is operable to test a plurality of different graphical user interfaces for a plurality of different software applications and wherein each software test tool is associated with a different tool-specific scripting language, that can be invoked to perform testing operations on the graphical user interface that is displayed while the software application is running, and wherein each of the plurality of different software test tools use only their associated tool-specific scripting language to test the plurality of different graphical user interfaces associated with a plurality of different software applications;

allowing a user to enter a test case input file stored on the computer readable medium, that contains a plurality of generic interface commands that are abstractions independent of any of the tool-specific scripting languages, wherein the test case input file can be edited and reused as necessary by the user to specify different generic interface commands for testing against a software application's graphical user interface in any of the different software test tools; and

using a plurality of dynamically loaded libraries corresponding to the plurality of different software test tools, and including [[a]] at least one library for each of the a plurality of different software test tools wherein each library is a group of functions written in each tool-specific scripting language, to receive the generic interface commands defined in the test case input file, identify which libraries are required, load the required libraries associated with the software test tool the user is currently using, map the generic interface commands to the software test tool's associated tool-specific scripting language, use the software test tool to perform the testing

operations on the software application's graphical user interface, including translating the generic interface commands to tool-specific commands, and report to the user the success or failure of the testing operations.

16-17 (Canceled).

18. (Previously Presented) The computer readable medium of claim 15 further comprising a rules-based wizard that guides the user to edit or create the test case input file by choosing the testing operations to be included in the test case input file wherein the rules-based wizard maps the testing operations to generic interface commands.

19. (Canceled).

20. (Previously Presented) The computer readable medium of claim 15 wherein the test case input file is created offline and subsequently communicated to the interpretive engine.

21. (Previously Presented) The computer readable medium of claim 15 wherein any of the test software tools can be removed and replaced with another test software tool.

22. (Previously Presented) The system of claim 1, wherein the system defines a contract interface for use as an entry point in loading the libraries corresponding to the plurality of different software test tools, and wherein additional software test tools that use a different scripting language can be dynamically plugged into the system at the entry point by defining an execution interface of those additional software test tools to comply with the contract interface.

23. (Previously Presented) The method of claim 8, further comprising defining a contract interface for use as an entry point in loading the libraries corresponding to the plurality of different software test tools, wherein additional software test tools that use a different scripting language can be dynamically plugged in at the entry point by defining an execution interface of those additional software test tools to comply with the contract interface.

24. (Previously Presented) The computer readable medium of claim 15 further comprising instructions which when executed cause the computer to perform the additional step of defining a contract interface for use as an entry point in loading the libraries corresponding to the

plurality of different software test tools, wherein additional software test tools that use a different scripting language can be dynamically plugged in at the entry point by defining an execution interface of those additional software test tools to comply with the contract interface.

25. (New) The system of claim 1 wherein each software test tool is used only for execution of the test case input file, and the test case input file is built independently of any software test tool.

26. (New) The system of claim 1 wherein a first tool-specific scripting language associated with a first software test tool is mapped to a second tool-specific scripting language associated with a second software test tool, enabling test cases written in the second tool-specific scripting language to be executed by the first software test tool.